IN THE CLAIMS:

Please cancel claims 16, 19, and 20, and amend the claims as follows:

Claim 1 (Currently Amended): Method for scheduling the service of a thread, said method comprising the steps of:

masking interrupts from ~~one or more~~ hardware devices in order to ignore interrupts for other threads;

acquiring a latency information associated with the thread, wherein the latency information indicates a time at which the thread needs to be processed;

unmasking interrupts from the ~~one or more~~ hardware devices in order to detect interrupts for the other threads; and

rearranging an order in which the thread and the other threads will be serviced in a single queue to schedule the thread for processing in accordance with said latency information, wherein the rearranging is performed simultaneously for the thread and the other threads, and the thread and the other threads that are ordered in the single queue correspond to all requests received from the hardware devices.

Claim 2 (Previously Presented): The method of claim 1, wherein said latency information is computed based on a buffer size or display rate.

Claim 3 (Previously Presented): The method of claim 1, further comprising computing the time at which the thread needs to be processed by summing the latency information with a current time.

Claim 4 (Previously Presented): The method of claim 1, wherein said latency information represents a time duration that is necessary to service the thread.

Claim 5 (Previously Presented): The method of claim 1, wherein said latency information represents a maximum time allowed before a first buffer will be emptied and a read operation will switch to process a second buffer.

Claim 6 (Previously Presented): The method of claim 1, wherein said latency information represents a time duration that is necessary to setup the thread to perform interrupt processing for the thread.

Claim 7 (Currently Amended): The method of claim 1, wherein said latency information is dependent on a hardware constraint for one of the ~~one or more~~ hardware devices.

Claim 8 (Previously Presented): The method of claim 1, wherein said latency information is provided by a device driver.

Claim 9 (Currently Amended): Apparatus for scheduling the service of a thread, said apparatus comprising:

means for receiving an interrupt from a hardware device;

means for masking interrupts from ~~one or more~~ hardware devices in order to ignore interrupts for other threads;

means for acquiring latency information associated with the interrupt, wherein the latency information indicates a time at which the thread needs to be processed;

means for unmasking interrupts from the ~~one or more~~ hardware devices in order to detect interrupts for the other threads; and

means for rearranging an order in which the thread and the other threads will be serviced in a single queue to schedule the thread to process the interrupt in accordance with said latency information, wherein the rearranging is performed simultaneously for the thread and the other threads, and the thread and the other threads that are ordered in the single queue correspond to all requests received from the hardware devices.

Claim 10 (Previously Presented): The apparatus of claim 9, further comprising computing the time at which the thread needs to be processed by summing the latency information with a current time.

Claim 11 (Currently Amended): The apparatus of claim 9, wherein said latency information is dependent on a hardware constraint for one of the ~~one or more~~ hardware devices.

Claim 12 (Original): The apparatus of claim 11, wherein said hardware constraint is a size of a buffer.

Claim 13 (Original): The apparatus of claim 11, wherein said hardware constraint is a fullness of a buffer.

Claim 14 (Previously Presented): The apparatus of claim 11, wherein said hardware constraint is dynamically computed based on a buffer size or display rate.

Claim 15 (Original): The apparatus of claim 9, wherein said latency information is generated by a device driver associated with the hardware device.

Claim 16 (Canceled)

Claim 17 (Previously Presented): The method of claim 1, further comprising toggling an interrupt line.

Claim 18 (Previously Presented): The method of claim 1, further comprising:
determining the thread should be activated; and
activating the thread for processing.

Claim 19 (Canceled)

Claim 20 (Cancelled)

Claim 21 (Currently Amended): The method of claim 1, further comprising:

creating the thread prior to the steps of masking, acquiring, unmasking, and rearranging when one of the ~~one or more~~ hardware devices is initialized, wherein the thread is created for use during processing of a first interrupt that the one of the ~~one or more~~ hardware devices is configured to generate; and

freeing the thread when the one of the ~~one or more~~ hardware devices is shut down.

Claim 22 (Currently Amended): The method of claim 21, further comprising:

creating an additional thread, wherein a first interrupt identification number is associated with the thread and a second interrupt identification number that is different than the first interrupt identification number is associated with the additional thread and the additional thread is created for use during processing of a second interrupt that the one of the ~~one or more~~ hardware devices is configured to generate; and

freeing the additional thread when the one of the ~~one or more~~ hardware devices is shut down.

Claim 23. (New): The method of claim 1, wherein the thread and at least one of the other threads correspond to interrupt requests from a single one of the hardware devices.